

REMARKS

Claims 1-9 and 11-19 are all the claims presently pending in the application. Claims 10 and 20 are canceled.

It is noted that Applicants specifically state that no amendment to any claim herein, if any, should be construed as a disclaimer of any interest in or right to an equivalent of any element or feature of the amended claim.

Claims 1-9 and 11-19 stand rejected under non-statutory double patenting over claims 1, 3-6, 8-12, and 14-19 of co-pending application S/N 10/671,937, further in view of “Superscalar GEMM-based Level 3 BLAS” to co-inventor Gustavson et al.

Claims 1-9 and 11-19 stand rejected under 35 U.S.C. § 112, first paragraph, as failing to comply with the written description requirement.

Claims 1-9 and 11-19 stand rejected under 35 U.S.C. § 112, second paragraph, as being indefinite.

Applicants understand that the previous rejection for claims 12-16 under 35 U.S.C. § 101, as directed to non-statutory subject matter, has been withdrawn.

Claims 1-20 stand rejected under 35 U.S.C. § 102(b) as anticipated by Gustavson et al.

These rejections are respectfully traversed in the following discussion.

I. THE CLAIMED INVENTION

The claimed invention is directed to a method of executing a linear algebra subroutine on a machine having at least one floating point unit (FPU) with one or more associated load/store units (LSU) to load data into and out of floating point registers (FRegs) of said FPU by way of a cache. For an execution code controlling an operation of the floating point unit (FPU) performing a linear algebra subroutine execution, instructions are inserted to move data into the cache providing data for the FPU so that the LSUs can move the data into the FRegs in a timely manner for the linear algebra subroutine execution. The data is prefetched into the cache from a memory in a nonstandard format predetermined to reduce a number of data streams for a level 3 processing to be three streams and to allow a multiple loading of

these streams into the FPU by the LSU, thereby improving an efficiency for the linear algebra subroutine execution

Conventional compilers do not have the capability to automatically pre-fetch (timely move) data into the FPU for Level 3 Dense Linear Algebra Subroutines, particularly in view of the newer architectures having FPUs and LSUs.

The claimed invention, on the other hand, teaches how to timely load data into cache, using a non-standard format predetermined to allow the minimum of three data streams and to allow multiple loading into the FPU. This feature can also be accomplished by conventional compilers, when modified to incorporate the concepts of the present invention.

II. THE DOUBLE PATENT REJECTION

The Examiner continues to consider that all pending claims of the present application are obvious over the claims of co-pending application 10/671,937.

In response, Applicants do indeed agree with Examiner Vicary that the present application is related to this co-pending application, but respectfully disagree with the Examiner that the claims constitute double patenting, for at least the following two reasons.

First, it is brought to the Examiner's attention that the present application relates to the pre-fetching of data into the cache system from main memory, whereas co-pending application 10/671,937 (YOR920030171US1) addresses the issue of bringing data from L1 cache into the registers of the FPU as a pre-loading operation of data for the processing executing in the FPU and that the pre-loading occurs simultaneous to the FPU processing.

Thus, although the two applications are quite related to the overall task of getting matrix data from memory into the FPU register set, they deal with two entirely different aspects of this task. The distinction is inherent in the meaning of "pre-fetching" versus "pre-loading", as those two terms are used for the present invention, wherein "pre-fetching" refers to the process of data retrieval from main memory into cache, and "pre-loading" refers to the process of loading data into the FPU registers from cache.

The claims are distinct when it is recognized that pre-fetching involves inserting instructions by the compiler so that data will be moved into cache as an operation related to the overall processing, whereas pre-loading involves instructions for controlling the FPU.

Moreover, this intended distinction between pre-fetching and pre-loading shows up in

the plain meaning of the claim language of the independent claims of the present application that refers to “inserting instructions to move data into said cache ...” into the “execution code controlling an operation of said floating point unit (FPU)....” In contrast, the independent claims of co-pending application 10/671,937 (YOR920030171US1) refers to “overlapping by preloading data into floating point registers”

Finally, it is noted that the non-standard format of the present invention, as well as those mentioned elsewhere in the six co-pending applications, involves more than any simple preloading or prefetching that might be found in the prior art, since it involves data structures predetermined to have the matrix data to be in a specific nonstandard format that overcomes deficiencies inherent either in the standard format and/or in various machine architectural deficiencies recognized by the present inventors and described appropriately in various of these co-pending applications.

For example, let the streaming data for L0 arrive in L1 so it is not contiguous for loading into L0. What is described in the cited co-pending application is to use the fast SIMD load k instruction anyway, but incorrectly. Now some new processors allow one to use load type instructions that are part of the L0 instruction set. Thus, by preloading L0 in advance, and then using L0 load instructions to correct the error of the L1 SIMD k load, the present inventions include another way to accomplish fast and efficient processing.

Thus, as a first point, Applicants respectfully submit that the two applications have wording that clearly distinguishes between the two concepts of pre-fetching versus “pre-loading.”

Second, Applicants respectfully disagree with the Examiner’s characterization that the present invention is obvious over co-pending application 10/671,937 (YOR920030171US1), further in view of the article “Superscalar GEMM-based” having co-inventor Gustavson as one of its co-authors.

That is, co-inventor Gustavson expressly states in this response that he and his co-authors did not address in that publication the claims in either the present application or co-pending application 10/671,937 (YOR920030171US1). The processors of the 1998 time period of that publication lacked the features of processors that were publicly announced in 2004 and later and for which the seven co-pending applications were addressed. These two new applications address these new features in the specific context of improving the performance of dense linear algebra (DLA) subroutines.

That is, relative to the Examiner's arguments in paragraph 5 beginning on page 3 of the Office Action, Applicants submit that newer processors have FPU's attached to their L1 caches. Data that enters L1 must be able to seamlessly enter the FPU, referred to by the present inventors as the L0 cache, meaning, more specifically, the register set of the FPU. It is no longer sufficient for the data merely to arrive in L1 for peak performance to occur, based on a hardware deficiency recognized by the present inventors for the L1/L0 interface of the newer architectures. Also, some newer processors bring data into L0 directly from L2 and thereby by-pass L1 cache. The two co-pending applications, in their respective claims, are addressing different aspects of these situations, and provide solutions that do not require that the hardware be re-designed for this deficiency.

Therefore, the Examiner is respectfully requested to reconsider and withdraw this rejection.

III. THE 35 USC §112, FIRST PARAGRAPH, REJECTION

Claims 1-9 and 11-20 stand rejected under 35 U. S.C. §112, first paragraph, as allegedly failing the written description requirement. It is first noted that, although Applicants do not agree with the Examiner's position about written description, Applicants believe that the addition into the present application of wording of the third and seventh co-pending applications, as per the above specification amendment, appropriately addresses any and all of the Examiner's concerns and can be incorporated in the present specification by reason of having incorporated by reference all of these co-pending applications.

More specifically, the Examiner is concerned about the terminology "non-standard format." In response, Applicants submit that non-standard format is well understood in the art dealing with dense linear algebra (DLA), as the follow definitions indicate. It is noted that these formal definitions are readily available and understood to one having ordinary skill in the art. For example, the "full" definition given below is readily available in Fortran and C manuals, and the "full" and "packed" definitions are available in the LAPACK guide or the ESSL guide.

Definitions of the two standard formats of matrices for Dense Linear Algebra:

Full definition: given M , N , $LDAR$ and $LDAC$. Let A be a Fortran array with $LDAR*LDAC$ locations; ie, $A(0:LDAR*LDAC-1)$. Let A be an M by N matrix residing in array A where $LDAR \geq M$ and $LDAC \geq N$. The $a(i,j)$ entry of A resides at location $A(i+LDAR*j)$ of array A , where $0 \leq i < M$ and $0 \leq j < N$. More precisely: $Loc(A(0)) + i + LDAR*j$.

Note: usually, $LDAC$ is not given; i.e., just $LDA=(LDAR)$, M , $N=(LDAC)$ are given, where $LDA \geq M$.

The Packed Definition: given N and $uplo = 'L'$ or $'U'$. A packed matrix is either a "symmetric" or a triangular matrix whose order is N . Order N AP resides in array $AP(0:nt-1)$; $nt=N*(N+1)/2$. For $uplo = 'L'$ we have $a(i,j)$ with $i \geq j$. $a(i,j)$ resides at $loc(AP(0)) + j*(N+N+1-j)/2 + i - j$. For $uplo = 'U'$ we have $a(i,j)$ with $i \leq j$. $a(i,j)$ resides at $loc(AP(0)) + j*(j+1)/2 + i$.

Description of the standard data format was presented in some detail on page 16 of co-pending application 10/671,935 (YOR920030330US1) and page 15 of co-pending application 10/671,934 (YOR920030331US1).

Moreover, the present application provides an indication of an even more stringent concept in the description at lines 12-15 of page 12:

"Additionally, in the present invention, it is preferable that the matrix data be laid out contiguously in memory in "stride one" form. "Stride one" means that the data is preferably contiguously arranged in memory to honor double-word boundaries and the useable data is retrieved in increments of the line size."

Moreover, for the methods of these two co-pending application to work at peak efficiency one needs to be using New Data Structures, a fact not yet recognized in the art prior to these seven co-pending applications. How the matrix data gets into and out of the caches greatly effects how efficient the DLA matrix subroutine will execute.

It is implicit in claim 1 of the present application that the matrix is stored in one of the two standard formats of DLA. In the paper "High-performance linear algebra algorithms using new generalized data structures for matrices", by Fred Gustavson, it is shown that the standard data structures hurt the performance of matrix subroutines. These inevitable results are generally accepted by the engineering and scientific community at large.

Broadly speaking, claim 1 refers to a better data structure (i.e., including "non-standard format") for matrices that the DGEMM subroutine will be processing using this invention. The result will be a more efficient execution of the DGEMM via the streaming

process. Prefetching (and preloading) will only work well for these better data structures. A non-standard data structure for DLA is any data structure that is not standard (as defined above), and the two non-standard data structures that are particularly relevant here are the ones described in, Composite Blocking, 10/671,887 (YOR920030010US1) and Register Block Data Format, 10/671,888 (YOR920030169US1).

Again, it is noted that these seven co-pending applications listed at the front of the present application are inter-related in the broader view of increasing of efficiency in matrix processing. They were separated into seven applications, anticipating an inevitable restriction requirement and necessity for six divisional applications that would have resulted in attempting to write one single disclosure and one set of claims for the seven different concepts.

The concept of streaming the three matrices is covered in more detail in co-pending application 10/671,934 (YOR920030331US1), with Figure 5 therein showing the streaming.

But it is also noted that streaming is implicitly described in the present application at various places, supporting the claim language of reducing the number of data streams to be three streams, at least as follows:

- Lines 1-5 of page 8: *“A key idea is that, whenever there are significantly more floating point operations than load/store operations (touches) in order to overcome (almost completely) the initial cost of bringing matrix operands A^T and, later, pieces of (swathes) B and (submatrix blocks) C .”*
- Lines 5-8 of page 9: *“At the core of level 3 Basic Linear Algebra Subprograms (BLAS) are “L1 kernel” routines which are constructed to operate at near the peak rate of the machine when all the data operands are streamed through or reside in the L1 cache.”*
- Lines 7-9 of page 13: *“The difference ($n^3 - n^2$) in the number of execution operations versus the number of loading operations allows for the prefetching of the data for the kernel routines.”*
- Line 21 of page 14 through line 2 of page 15: *“Dimension N will be defined as the streaming dimension. “Streaming” is the concept in which one matrix is considered resident in the L1 cache and the remaining two matrix operands (called “streaming matrices”) reside in the next higher memory level(s), e.g., L2 and L3. The streaming dimension is typically large.”*
- Lines 6-13 of page 15: *“There are three matrices A , B , C to be prefetched using the guiding principle. Figure 1 illustrates the case wherein A is the L1 cache resident matrix (i.e., in L1). Therefore, the DGEMM kernel subroutine DATB will*

need: a) "All" of the A^T (an almost L1-sized block). For consideration of the kernel operands only, the matrix size $M \times K$ elements will suffice. b) A column swath of $B(K \times NB$ elements). c) A register block of $C(MB \times NB$ elements)."

- Lines 2-5 of page 17: *"It must be determined when the matrix (matrices) under consideration must be touched. "Touching" is a term well understood in the art as referring to accessing data in memory in anticipation of the need for that data in a process currently underway in the processor."*

Therefore, Applicants submit that even the present application supports the claim language that there are three streams of matrix data.

Relative to the Examiner's concern in paragraph 29 concerning the terminology "allow a multiple loading of loads", it is first noted that this claim wording has been modified. However, Applicants submit that the plain meaning of this earlier terminology is not necessarily the Examiner's interpretation directed to the capability of the LSU to load multiple registers with one instruction, although such capability is indeed present in the newer computers to which the present invention is directed. Rather, this wording is intended to convey that the LSU will be able to perform loading a multiple number of times, as based the inserted instructions. Thus, the description at lines 7-9 of page 13 recites: *"The difference ($n^3 - n^2$) in the number of execution operations versus the number of loading operations allows for the prefetching of the data for the kernel routines."*

This wording is further supported by the description at lines 6-8 of page 14: *"The guiding principle is "functional parallelism." That is, the load/store unit(s) (LSUs) and the FPU(s) can be considered to be independent processes/processors."*

However, to more appropriately address the Examiner's specific interpretation of this feature on newer architectures, one example of this capability of "multiple loads" of this architecture is explained in co-pending application 10/671,888 (YOR920030169US1), wherein is described a crisscross loading of two data words into the FPU registers, which method is used to commit a "second error" that, together with a "first error" of using the non-standard format, referred to in that co-pending application as a "pseudo matrix", allows the combination of the non-standard pseudo matrix format plus the subsequent crisscrossing double loading operation of data into the FPU registers to overcome a hardware deficiency at the L1/FPU interface.

Newer processors, such as those beginning in early 2004, allow the use of a load multiple instruction or a SIMD load as a new type of instruction. Thus, in a single cycle,

multiple (k) data elements can be transferred from L1 into k consecutive floating point registers in L0. k is a small integer, typically 2 or 4. The SIMD load instruction only applies to data elements in L1 or L2 or elsewhere if the data items are consecutive and properly aligned. When this is not the case, performance can suffer by a factor of k. The cited co-inventor's paper dealt with multiple loads where each load had a $k = 1$, since the newer instruction capability did not exist at that time.

Whatever wording the Examiner might be able to find in this article has to be considered as being out-of-context of this newer capability. For those processor types there was no L1, L0 seamless problem, as any L1 element could be loaded in a *single* cycle into a *single* floating register. The present inventors discovered that this interface was no longer seamless and developed various software methods to address this problem, rather than having the newer architecture redesigned with hardware modifications.

Relative to the Examiner's concern in paragraph 30 concerning support for the wording "existing in a Level 3 Dense Linear Algebra Subroutine" in claims 2, 11, and 13, Applicants bring the Examiner's attention to the description at lines 7-9 of page 13 ("*The difference ($n^3 - n^2$) in the number of execution operations versus the number of loading operations allows for the prefetching of the data for the kernel routines.*"), noting that the claim language actually refers to "time slots" in the subroutine. Thus, Applicants submit that the specification does indeed support wording based upon unused time slots in existing subroutines that could be used for inserting additional move instructions that could be used for the prefetching of the present application.

In view of the above, Applicants respectfully submit that the specification does indeed support the claim wording of the claimed invention. Therefore, the Examiner is respectfully requested to reconsider and withdraw this rejection.

IV. THE 35 USC §112, SECOND PARAGRAPH, REJECTION

Relative to the Examiner's concern in paragraph 32 for the terminology "timely moved" or "timely manner", Applicants believe that the definition recited for "touching" at the top of page 17 adequately addresses this concern: "*Touching* is a term well understood in the art as referring to accessing data in memory in anticipation of the need for that data in a process currently underway in the processor."

Relative to the Examiner's concern in paragraph 33, Applicants believe they have addressed this issue by giving above the definition of "non-standard format."

Relative to the Examiner's concern in paragraph 34, Applicants submit that "Level 3 processing" is a commonly-used term by the DLA community. It means doing $O(n^3)$ operations on $O(n^2)$ data. Take three DO or three FOR loop that run from 1 to n. Simple counting shows that there exactly n^3 values of i, j, k. Therefore, Level 3 means a code that has three DO or FOR loops.

Relative to the Examiner's concern in paragraph 35, Applicants consider that this is a good observation but believe that the responses above hopefully answer this query.

Relative to the Examiner's concern in paragraph 36, Applicants believe that the above amendments to claim 6 appropriately address the Examiner's request for clarification. It is matrix elements, in chunks of lines, that are being moved from memory into L2 / L1 / L0 from memory. "Moving instructions" cause data outside of a cache to be fetched into cache.

In view of the foregoing, the Examiner is respectfully requested to reconsider and withdraw this rejection.

V. THE PRIOR ART REJECTION

The Examiner alleges that the article "Superscalar GEMM-based Level 3 BLAS –The On-going Evolution of a Portable and High-Performance Library," Para '98, pages 207-215), co-inventor Gustavson, et al., teaches the claimed invention.

In response, Applicants submit that this paper refers only to multiple loads of load multiple type $k=1$. The present application addresses architectures capable of a SIMD load with $k > 1$, and this paper by Gustavson et al., does not cover the specific situation that the present invention can address for the newer architectures.

Moreover, the independent claims of the present application also refer to non-standard format used when pre-fetching data from memory into L1 cache. This aspect of independent claim 1 is not present in this paper. In paragraph 3 on page 12 of the Office Action, the Examiner relies upon the description in section 3.1, first indented paragraph of page 210 of this paper, presumably meaning the following:

"This technique, to keep a small square block of C in registers and replace entries of

A and B between consecutive iterations of the innermost loop, maximizes the ratio between the number of MAAs and the number of load and store instructions, used to transfer data to and from registers, i.e., #MAAs/(#LOADs + #STOREs) is maximized.”

However, Applicants respectfully disagree with the Examiner that one having ordinary skill in the art would agree with the Examiner’s characterization that this description has anything at all to do with data format, let alone data format used for data moved into L1 cache. Indeed, Applicants respectfully submit that these words have no suggestion whatsoever as to whether the data anywhere in this sentence is in any specific format, since the mechanism is not described as dependent upon any such specific data format.

Therefore, Applicants submit that there is no suggestion in co-inventor Gustavson’s cited paper concerning non-standard format for data transfer between memory and cache, as required by the plain meaning of the claim language.

Hence, turning to the clear language of the claims, in Gustavson there is no teaching or suggestion of: “... inserting instructions to move data into said cache providing data for said FPU so that said LSUs can move said data into said FRegs in a timely manner for said linear algebra subroutine execution, said data being prefetched into said cache from a memory in a nonstandard format predetermined to reduce a number of data streams for a level 3 processing to be three streams and to allow a multiple loading of these streams into said FPU by said LSU”, as required by independent claim 1.

Relative to the Examiner’s comments in paragraph 6 on page 16 of the Office Action, relative to claims 2, 11, and 13, as explained above, the present invention differs from the discussion in Gustavson in that the computer architectures addressed by the present invention includes the capability that the pre-fetched data be able to enter L0 via SIMD load k instruction, where $k > 1$. This concept is not present in the Gustavson paper.

Relative to the Examiner’s comments in paragraph 38 on page 17 that there is no third matrix being streamed, Applicants submit that the third matrix in the example is matrix C.

That is, at any given time instant, a tiny submatrix of C, called a register block (RB) in the exemplary embodiment described in more detail elsewhere, resides in L0. Each element of the RB is being accumulated into via fused multiply add instructions ($c = c - a*b$). In this exemplary example, the *a* and *b* operands are from the A and B matrices, which are being streamed into L0. The A, B matrices is preferably in RB format for the method of the present application to work at peak efficiency. In a DLA factorization algorithm of a matrix D, say

PD = LU, the A, B, C here are submatrices of D. Some analysis will show that a particular C is also playing the role of the A and B streaming matrices during other time instances.

Relative to the Examiner's comment in paragraph 39 on page 17 of the Office Action, Applicants note that, technically speaking, this is an astute and valid remark by the Examiner. The technical support for recognizing that concepts of the seven co-pending applications apply generally for all level 3 subroutines can be found in the previously-mentioned article by co-inventor Gustavson ("High-performance linear algebra algorithms using new generalized data structures for matrices"), wherein it is shown that these concepts apply generally for level 3 subroutines.

However, relative to the present application itself, the concept that the present invention applies generally is found in the original application at least in noting that the original claims included this description. Moreover, Applicants submit that the description at lines 17-19 of page 12 ("*The present invention lowers the cost of the initial, requisite loading of data into the L1 cache for use by Level 3 kernel routines in which the number of operation steps are of the order of n^3 .*") and at lines 10-11 of page 13 ("*It is sufficient to describe the implementation of the present invention as it relates to the BLAS Level 3 DGEMM L1 cache kernel.*")

Thus, Applicants respectfully submit that the original disclosure does indeed support the claim language that the present invention covers any Level 3 subroutine.

Therefore, Applicants submit that all elements of the claimed invention were adequately described in the originally-filed application and that there are elements of the claimed invention that are not taught or suggested by this earlier publication by Gustavson, and the Examiner is respectfully requested to withdraw this rejection.

VI. FORMAL MATTERS AND CONCLUSION

In view of the foregoing, Applicant submits that claims 1-9 and 11-19, all the claims presently pending in the application, are patentably distinct over the prior art of record and are in condition for allowance. The Examiner is respectfully requested to pass the above application to issue at the earliest possible time.

Should the Examiner find the application to be other than in condition for allowance, the Examiner is requested to contact the undersigned at the local telephone number listed

Serial No. 10/671,889
Docket No. YOR920030170US1 (YOR.464)

below to discuss any other changes deemed necessary in a telephonic or personal interview.

The Commissioner is hereby authorized to charge any deficiency in fees or to credit any overpayment in fees to Assignee's Deposit Account No. 50-0510.

Respectfully Submitted,



Date: January 27, 2008

Frederick E. Cooperrider
Registration No. 36,769

McGinn Intellectual Property Law Group, PLLC
8321 Old Courthouse Road, Suite 200
Vienna, VA 22182-3817
(703) 761-4100
Customer No. 21254

CERTIFICATION OF TRANSMISSION

I certify that I transmitted electronically, via EFS, this Amendment under 37 CFR §1.111 to the USPTO on January 27, 2008.



Frederick E. Cooperrider (Reg. No. 36,769)